Fakulta aplikovaných věd
Západočeská univerzita v Plzni

Překladač vlastního jazyka

KIV/FJP

Martin Brožek
A20B0116P
Jaroslav Hrubý
A21N0046P

5. 2. 2023

Obsah

Kapitola 1

Zadání

Cílem práce je vytvoření překladače zvoleného jazyka. Je možné inspirovat se jazykem PL/0, vybrat si podmnožinu nějakého existujícího jazyka nebo si navrhnout jazyk zcela vlastní. Dále je také potřeba zvolit si pro jakou architekturu bude jazyk překládán (doporučeny jsou instrukce PL/0, ale je možné zvolit jakoukoliv instrukční sadu pro kterou budete mít interpret).

Jazyk musí mít minimálně následující konstrukce:

- definice celočíselných proměnných
- definice celočíselných konstant
- přiřazení
- základní aritmetiku a logiku (+, -, *, /, AND, OR, negace a závorky, operátory pro porovnání čísel)
- cyklus (libovolný)
- jednoduchou podmínku (if bez else)
- definice podprogramu (procedura, funkce, metoda) a jeho volání

Dále je možnost jazyk doplnit o některé z následujících rozšíření:

Jednoduchá rozšíření (1 bod):

- každý další typ cyklu (for, do .. while, while .. do, repeat .. unitl, foreach pro pole)
- else větev
- datový typ boolean a logické operace s ním
- datový typ real (s celočíselnými instrukcemi)
- datový typ string (s operátory pro spojování řetězců)
- rozvětvená podmínka (switch, case)
- násobné přiřazení (a = b = c = d = 3;)
- podmíněné přiřazení / ternární operátor (min = (a ¡ b) ? a : b;)
- paralelní přiřazení (a, b, c, d = 1, 2, 3, 4;)
- příkazy pro vstup a výstup (read, write - potřebuje vhodné instrukce které bude možné využít)

Složitější rozšíření (2 body):

- příkaz GOTO (pozor na vzdálené skoky)
- datový typ ratio (s celočíselnými instrukcemi)
- složený datový typ (Record)
- pole a práce s jeho prvky
- operátor pro porovnání řetězců
- parametry předávané hodnotou
- návratová hodnota podprogramu
- objekty bez polymorfismu
- anonymní vnitřní funkce (lambda výrazy)

Rozšíření vyžadující složitější instrukční sadu než má PL/0 (3 body):

- dynamicky přiřazovaná paměť - práce s ukazateli
- parametry předávané odkazem
- objektové konstrukce s polymorfním chováním
- instanceof operátor
- anonymní vnitřní funkce (lambda výrazy) které lze předat jako parametr
- mechanismus zpracování výjimek

Kapitola 2

Vlastnosti jazyka

Náš jazyk dostal pracovní název *AdAs*. Jeho překladač jsme se rozhodli implementovat v programovacím jazyce *Java* za pomocí nástroje *ANTLR*, který dokáže na základě definované vstupní gramatiky vytvořit lexikální, syntaktický nebo sémantický analyzátor. Cílovou platformou našeho jazyka je *PL/0*, respektive její instrukční sada.

Zvolená rozšíření pro náš jazyk jsou:

- další typ cyklu - while, do-while, for-each
- else větev
- datový typ boolean a logické operace s ním
- rozvětvená podmínka (switch, case)
- pole a práce s jeho prvky
- parametry předávané hodnotou
- návratová hodnota podprogramu

Ze zadání a námi zvolených rozšíření vyplývá, že jazyk bude umět pracovat s proměnnými datového typu `boolean` (pravdivostní hodnota) a `int` (celé číslo). Mezi těmito typy bude implicitní konverze. Pokud je číselná hodnota různá od 0, je při jakémkoliv použití v roli pravdivostní hodnoty převedena na hodnotu `true`. Pro volání funkcí, aritmetické a logické operace jsme se rozhodli využít prefixovou notaci, viz sekce Jazykové konstrukce.

Kapitola 3

Popis implementace

3.1 Gramatika

Jak bylo již zmíněno, vstupem nástroje *ANTLR* je definovaná gramatika. Tato gramatika musí být napsána v rozvinuté Backusově–Naurově formě (EBNF). Takto napsaná gramatika pak určuje jazyk a na výstupu vytváří zdrojový kód, který umí daný jazyk rozpoznávat. Naše gramatika je uvedena klíčovým pravidlem `program`. Toto vrcholové pravidlo obsahuje veškerá další pravidla pro definici funkcí, podmínek, cyklů, proměnných nebo volání již definovaných funkcí. Každá funkce musí být definována před jejím zavoláním jinde v programu. Gramatika dále vynucuje přítomnost hlavní funkce, která nese název `main`.

3.2 Jazykové konstrukce

V našem jazyce lze vytvářet následující konstrukce:

- definice funkce

  ```
  function <návratová_hodnota> <název_funkce>(<parametry>) {

  <tělo_funkce>

  }
  ```

- volání funkce

  ```
  (<název_funkce>, <parametry>);
  ```

- definice proměnné

  ```
  <název_proměnné> <datový_typ>;
  ```

- definice konstanty

  `const <název_konstanty> <datový_typ>;`

- definice proměnné pole

  `<název_proměnné>[<velikost_pole>] <datový_typ>;`

- přiřazení

  `<název_proměnné> = <hodnota>;`

  `<název_pole>[<index>] = <hodnota>;`

- aritmetické operace

  `(+, a, b)`

  `(-, a, b)`

  `(*, a, b)`

  `(/, a, b)`

- logické operace

  `(AND, a, b)`

  `(OR, a, b)`

  `(!, a)`

- cykly

  `while (<podmínka>) { <tělo> }`

  `do { <tělo> } while (<podmínka>)`

  `for <identifikátor> in <identifikátor_pole> { <tělo> }`

  `for <identifikátor> in <limit | identifikátor> { <tělo> }`

- podmínky

```
if (<podmínka>) { <tělo> } else { <tělo> }

switch (<identifikátor>) {
case <hodnota>: { <tělo> }
default: { <tělo> }
}
```

3.3 Průběh překladu

Před začátkem samotného překladu zkontrolujeme předané parametry. Pro bezchybný průběh překladu jsou zapotřebí dva vstupní parametry. Prvním parametrem je název vstupního souboru, kde se nachází zdrojový kód v našem programovacím jazyce *AdAs*. Druhým parametrem je pak výstupní soubor, kam budou zapsány výsledné instrukce *PL/0*.

Po ověření správnosti parametrů jsou inicializovány potřebné struktury a spuštěn překlad. Samotný překlad má dvě pomyslné fáze. V první fázi procházíme syntaktický strom, jež nám zprostředkuje nástroj *ANTLR*. Při procházení se celý program plní do námi připravených objektů, které reflektují gramatiku jazyka. V této fázi navíc kontrolujeme všechny nevalidní stavy, které by mohli být zapříčiněny chybou ve zdrojovém kódu, například přístup za hranice pole. Po kontrole zdrojového kódu a naplnění objektů může překladač přejít do druhé pomyslné fáze. Tou je zápis instrukcí *PL/0* podle nalezených jazykových konstrukcí ve vstupním zdrojovém kódu. Po úspěšném překladu jsou všechny instrukce zapsány do výstupního souboru.

Kapitola 4

Závěr

Výsledkem naší semestrální práce je funkční překladač pro náš programovací jazyk *AdAs*. V tomto jazyce lze pohodlně vytvářet jednoduché jazykové konstrukce, provádět aritmetické či logické operace nebo definovat vlastní funkce, které lze následně i volat. Díky této semestrální práci jsme se blíže seznámili s problematikou překladačů a vyzkoušeli si práci s nástrojem *ANTLR*. V neposlední řadě jsme se naučili, jak funguje nízkoúrovňový jazyk *PL/0*.